Ethan Chen

Operating System

I have the following Classes:

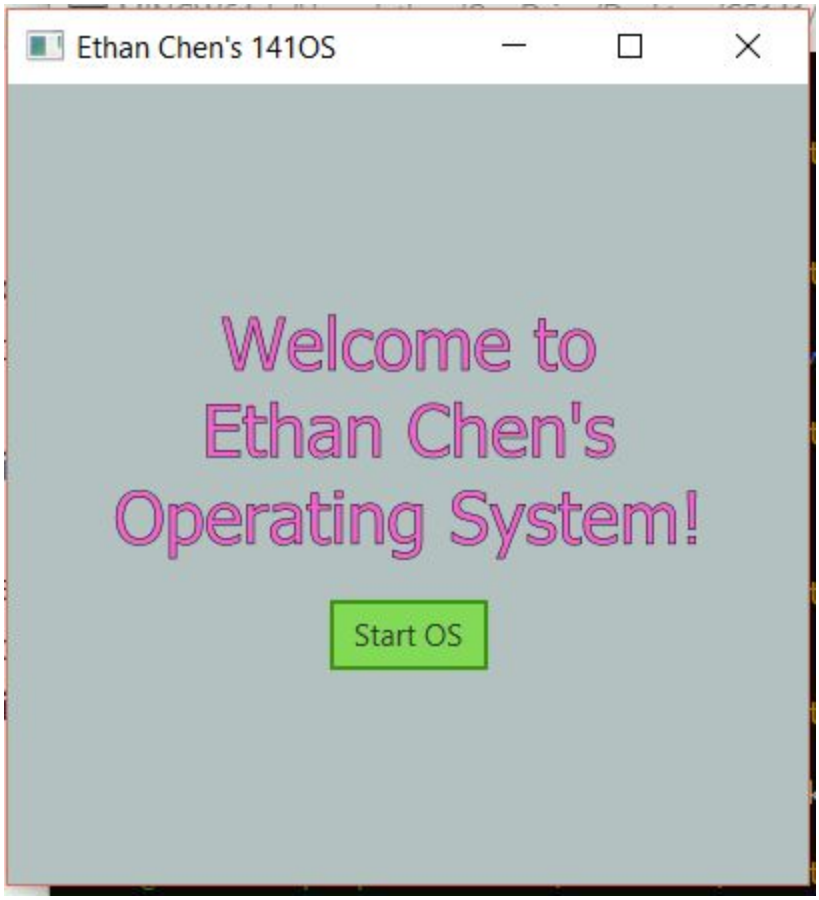
- DriverClass2 -
    - This is my main public class that handles the creation of my GUI and also implements the beginning of my Operating System by starting all of my **UserThreads**.
- UserThread -
    - This is my Thread class that controls a user. The UserThread reads commands from its associated input file, and evaluates them:
        - Save: UserThread attempts to check out an available **Disk** by requesting one from the **Disk ResourceManager**
        - Once a Disk is checked out by this UserThread, the user writes each line from its file into this Disk at its current Index. The current Index is given by the **DiskManager**.
        - End: Upon reaching an end in the input file, the Disk that was being written to is released through the **Disk Resource Manager**.
        - Print: The UserThread creates a new **PrintJobThread** to handle the printing of the given filename.
- PrintJobThread -
    - This is the Thread class that handles the printing of a specified file
    - The PrintJobThread requests an open **Printer** from the **Printer Resource Manager**
    - The printJobThread **reads** a line from the **Disk** at the correct Index specified by the **FileInfo** stored in the **DirectoryManager**'s hashTable.
    - The PrintJobThread then sends the StringBuffer it just read to the Printer it successfully requested.
- Printer
    - The printer receives StringBuffers from a PrintJobThread, then writes it into its associated output file.
- Disk
    - Can store data upon a write from a userThread
    - Can retrieve data for printJobThreads requesting it.

Concurrency is managed because each of my Users are separate Threads. Furthermore, UserThreads handling each print command by creating an additional PrintJobThread allows the user to keep running while a print is going. This potentially allows multiple print jobs running at the same time from one user.

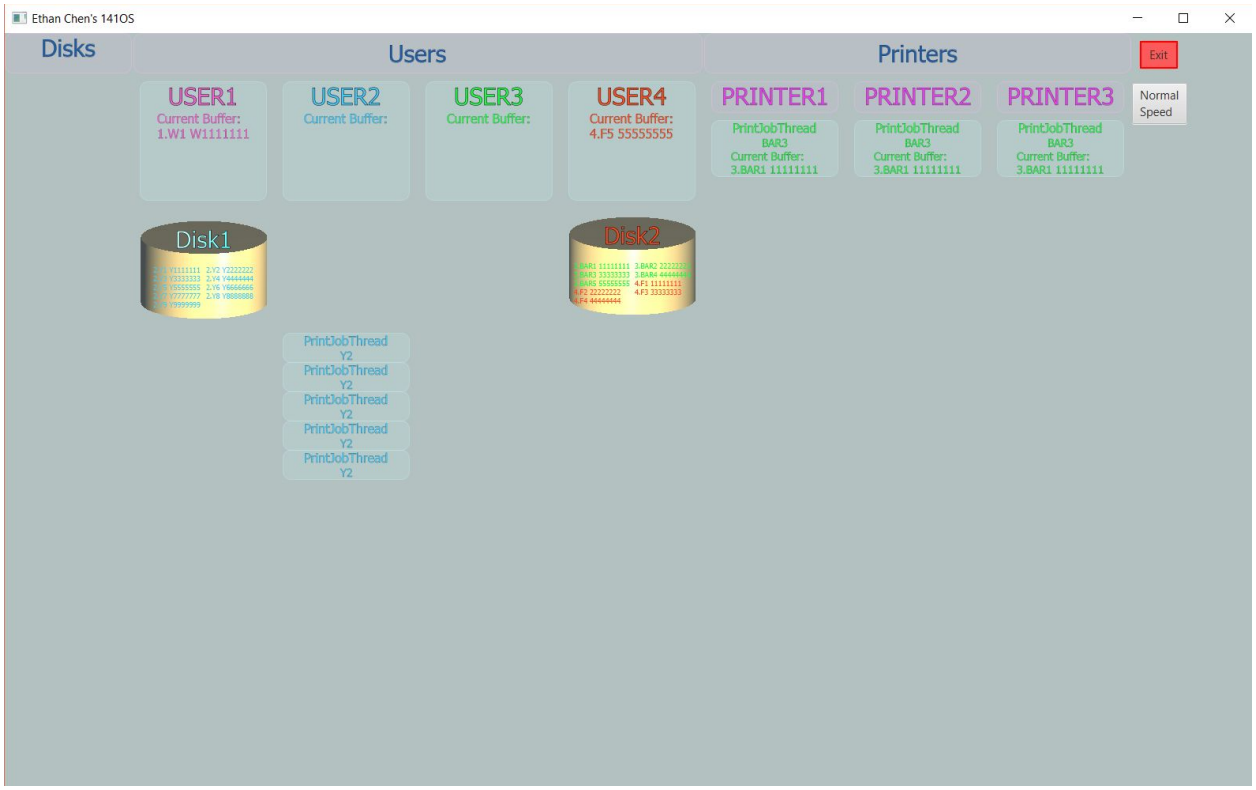
Synchronized means that the function Request will not continue until gets notified to do so. This pauses a Thread that makes a request. For example, Userthreads cannot continue reading

commands after “Save” until there is a released Disk for them to check out. In this example, release() notifies Request to continue;

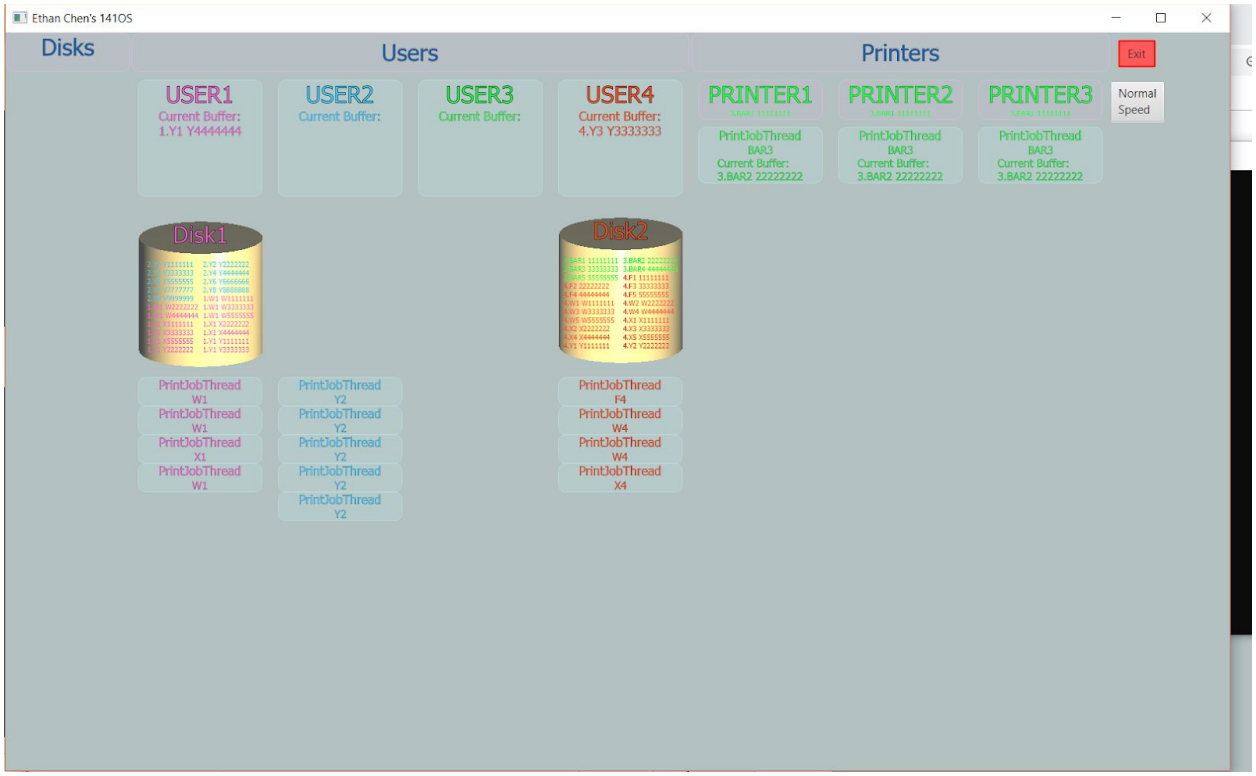
Welcome Screen:



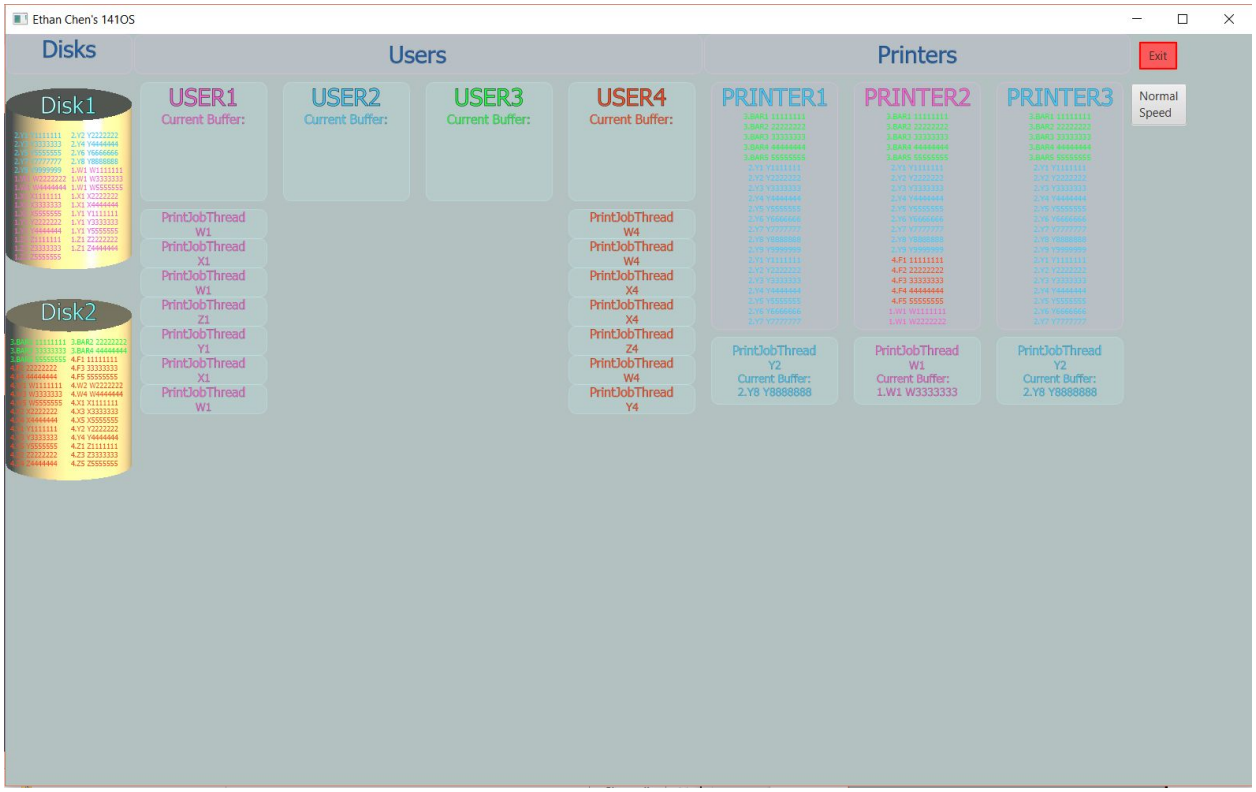
Users Starting to Read Commands



User3 is done reading commands: Others continue



Users Done Reading Commands:



Final State: